IBM Software Group

# Achieving Agility at Scale
# Improving Software Economics

**Walker Royce**
*Vice President, Rational Worldwide Services*

Rational. software


© 2009 IBM Corporation

# Agenda

- **Managing Variance: the Value of Agile Methods**
- Controlling Software Economics Through Measurement
- IBM's Transition to Agility @ Scale
- Some Final Thoughts

# Software development obsolesced by software delivery

| Software Development | Software Delivery |
| --- | --- |
| *Distinct development phase* | ***Continuously evolving systems*** |
| *Distinct handoff to maintenance* | ***No distinct boundary between development and maintenance*** |
| *Requirements-design-code-test sequence* | ***Sequence of released capabilities with ever increasing value*** |
| *Phase and role specific tools* | ***Common platform of integrated process / tooling*** |
| *Collocated teams* | ***Distributed, web based collaboration*** |
| *Standard engineering governance* | ***Economic governance tailored to risk / reward profiles*** |
| *Engineering practitioner led* | ***Business value and outcome led*** |

# Critical culture shifts in improving software economics

## *Conventional Governance*

**Activity-based management**
- Mature processes, PMI/PMBOK
- Plan in detail, then track variances

**Adversarial relationships**
- Paper exchange, speculation

**Requirements first**
- Assumes certainty in desired product
- Avoid change

**Early false precision**
- "More detail = higher quality"

**Apply too much or too little process**
- Process is primary, blind adherence

## *Agile Governance*

**Results-based management**
- More art than engineering
- Plan/steer/plan/steer…

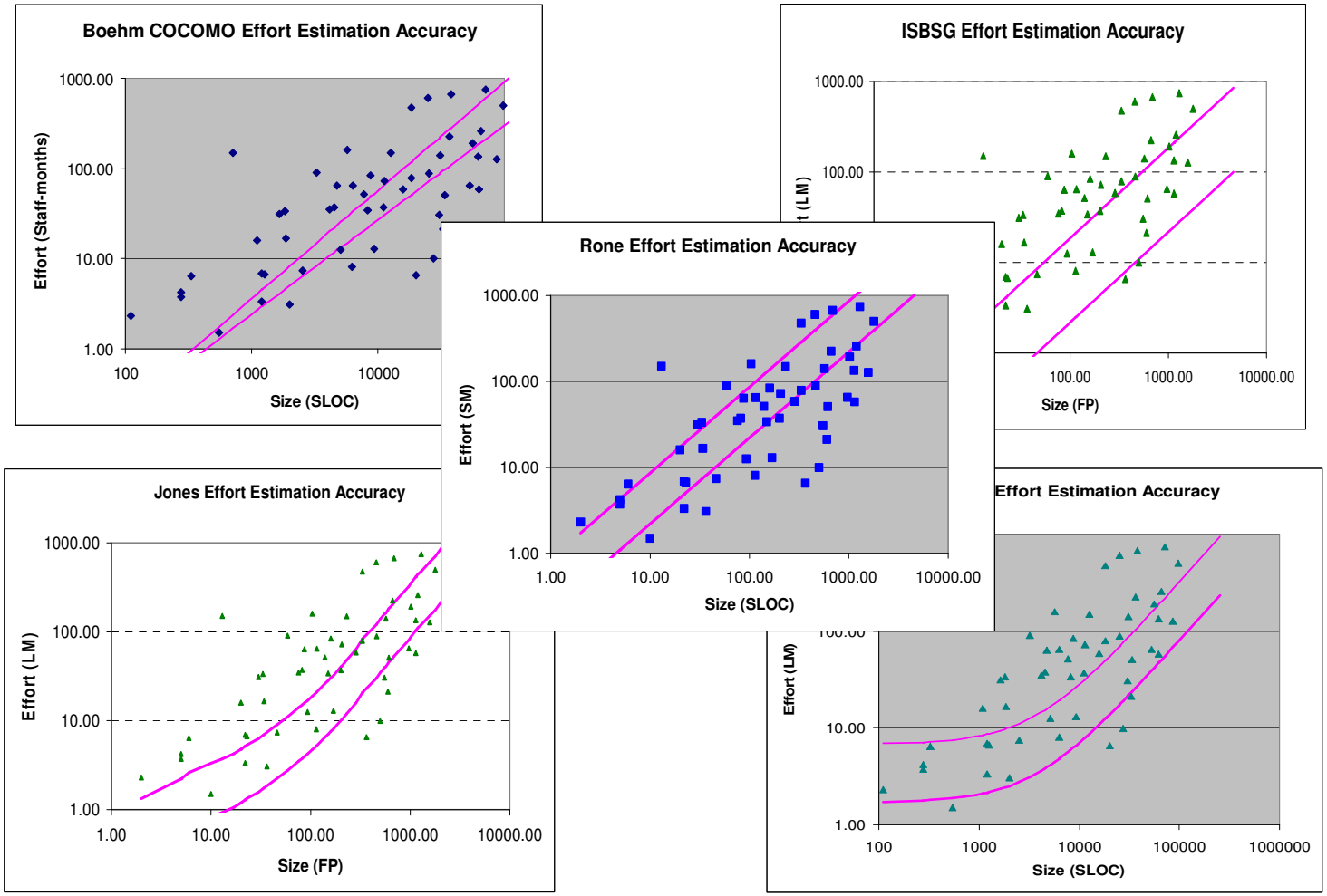
**Honest collaborative communication**
- Progressions/digressions, facts

**Architecture (*risk mitigation*) first**
- Admits uncertainties
- Manage change

**Evolving artifacts**
- Scope (Problem specs)
- Design (Solution specs)
- Constraints (Planning specs)

**Right-size the process**
- Desired results drive process
- Manage variances

# Software cost models




From George Stark, Paul Oman, “A comparison of parametric Software Estimation Models using real project data”, in press

# Improving software economics

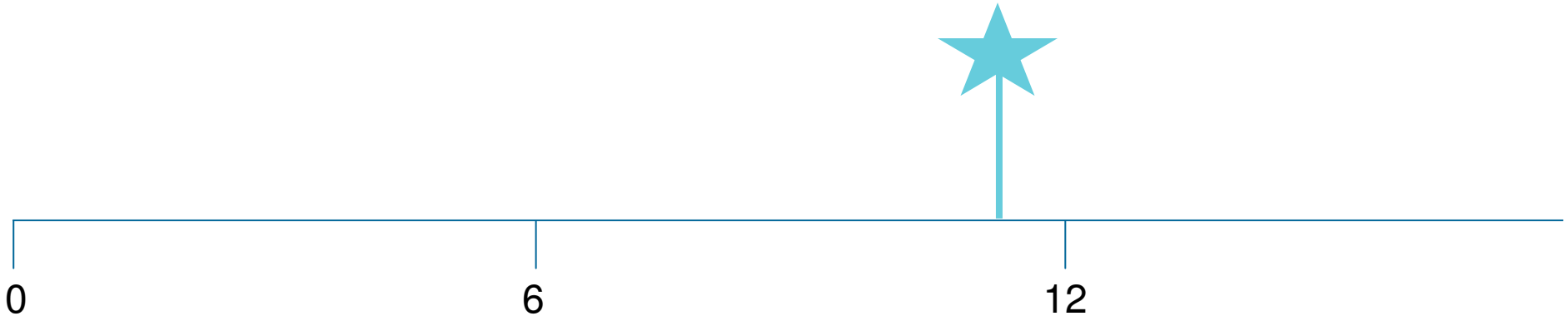
- Empirical software cost estimation models for:
  - Enterprise modernization, software maintenance
  - New developments, new releases, early prototypes
  - Packaged applications, systems engineering

**Time or Cost To Build = (Complexity) $^{(Process)}$ * (Team) * (Tools)**

| Complexity | Process | Team | Tools |
|---|---|---|---|
| ▪ Volume of human generated stuff<br>- KSLOC, FPs, UCs<br>▪ Quality/performance<br>▪ Scope | ▪ Methods<br>▪ Maturity<br>▪ Agility<br>▪ Precedence | ▪ Skills/Experience<br>▪ Collaboration<br>▪ Motivation | ▪ Automation<br>▪ Process enactment |

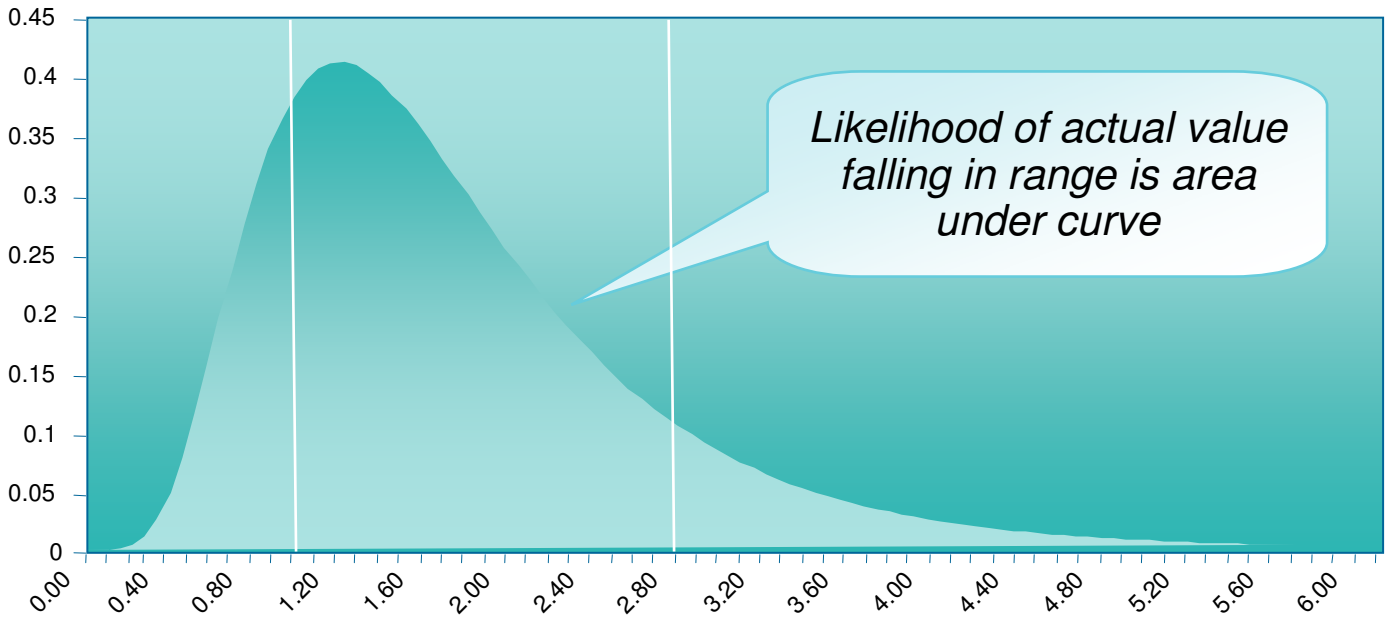
# Schedule risk: Imagine you have 12 months to deliver a business critical system

- Your estimators tell you it will be done in 11 months
- What do you do with the information?
  - Rest easy, believing there is no risk?



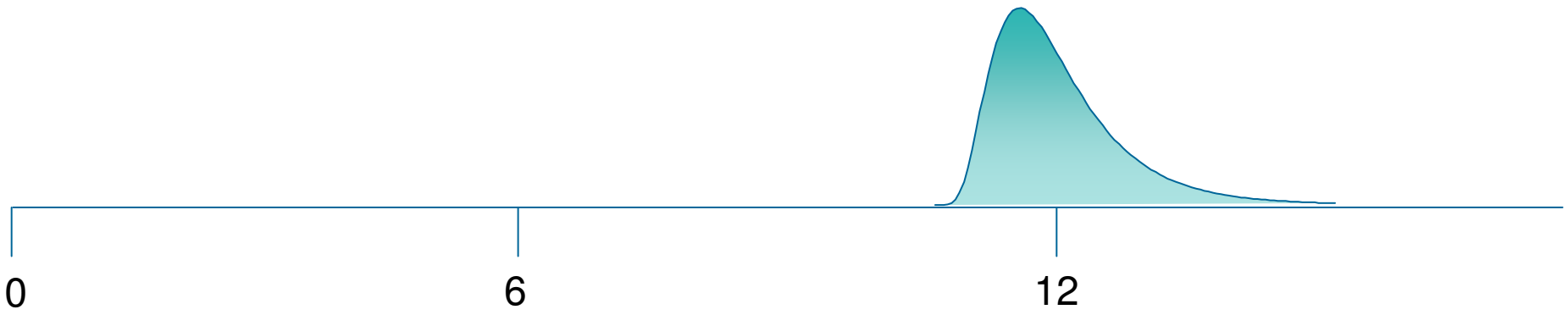
# Maybe you realize that program parameters (cost, schedule, effort, quality, …) are random variables

- Area under curve describes probability of measurement falling in range



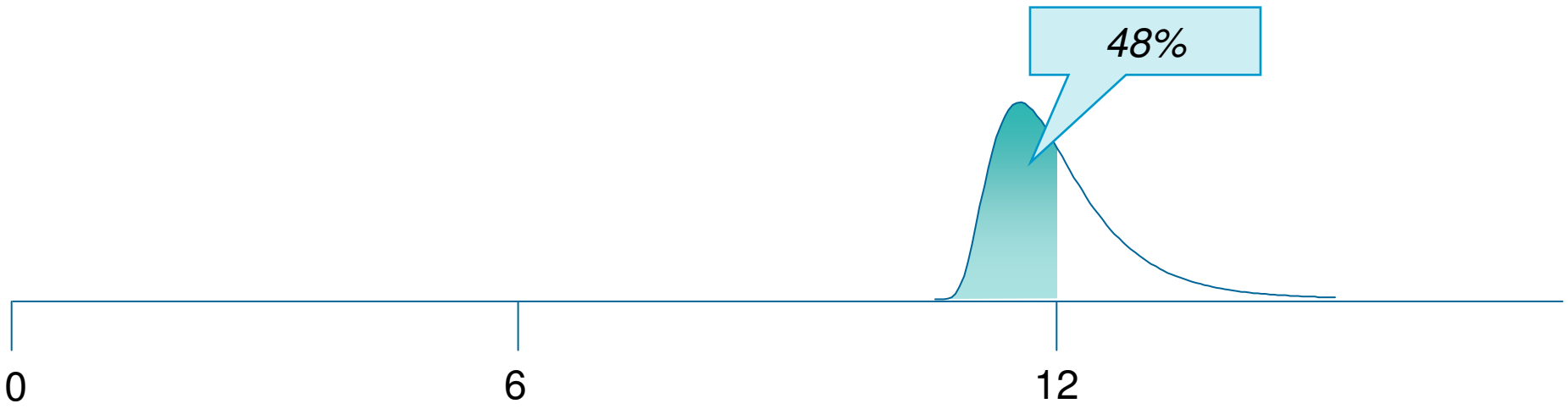
# Imagine you have 12 months to deliver a business critical systems

- So you ask for the distribution and discover there is some uncertainty



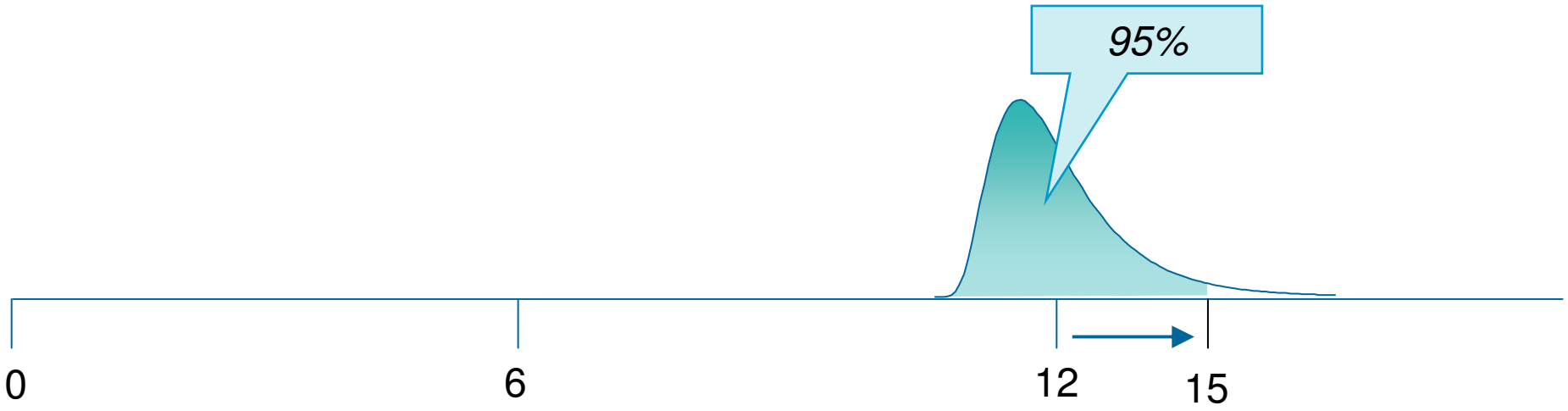
# Imagine you have 12 months to deliver a business critical systems

- In fact there is less than 50% chance of making the date



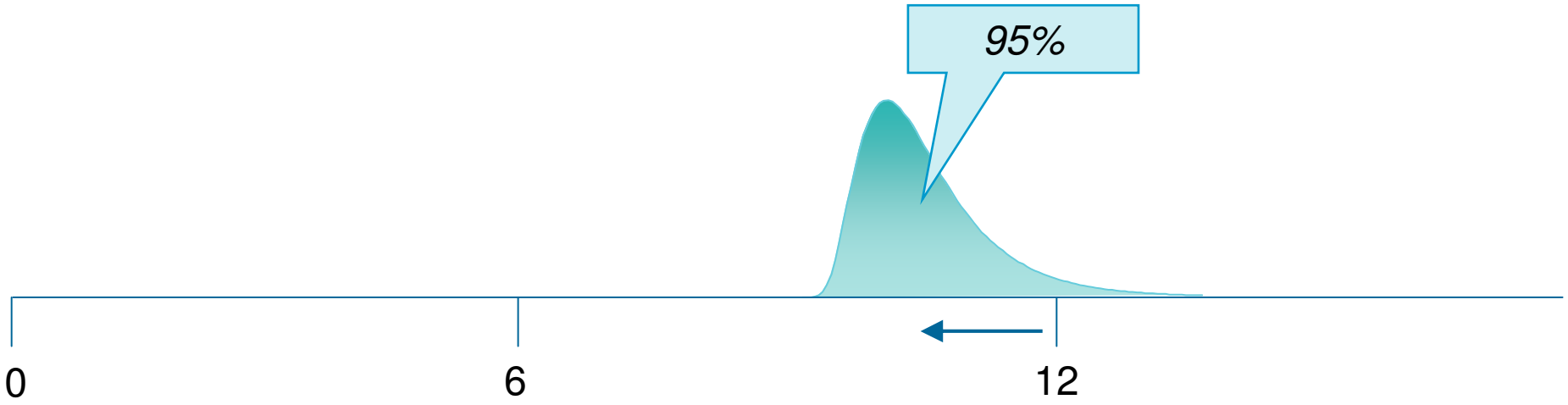
# Then what?

- Move out the date to improve likelihood of shipping?



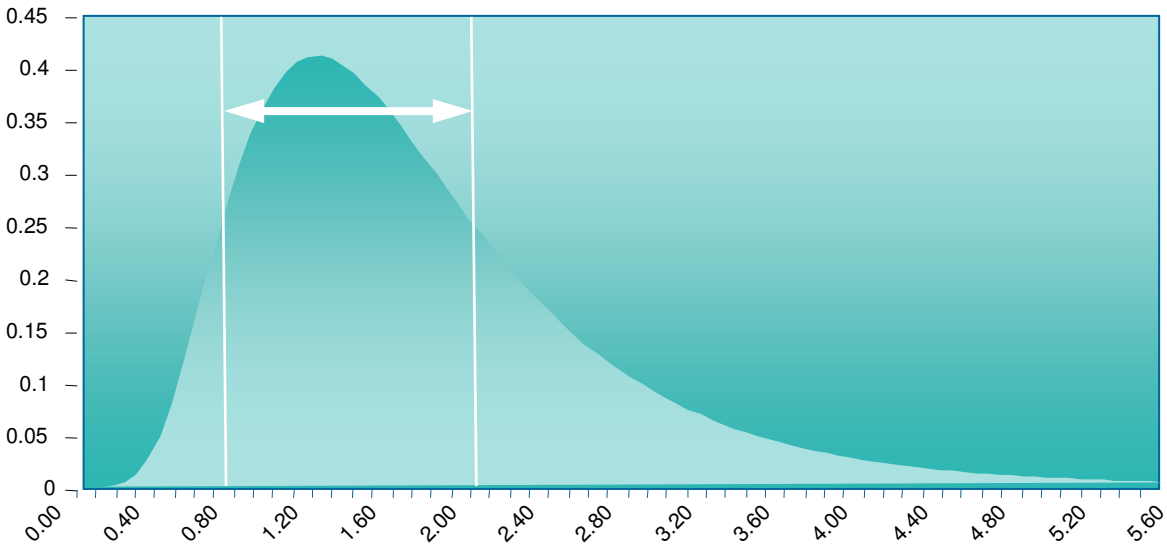
# Then what?

- Or move in the estimate by sacrificing quality or content?



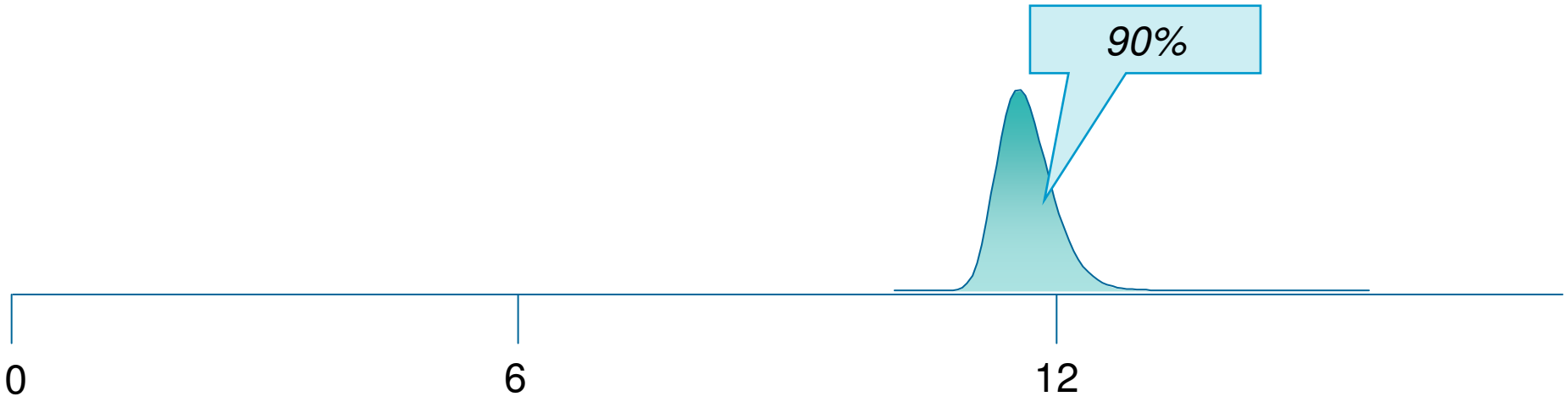
# Managing variances in scope, solution, plans: The real key to improving software economics

- Sources of uncertainty and variance
  - Lack of knowledge
  - Lack of confidence
  - Lack of agreement
- Reduction of variance reflects
  - Increased predictability of outcome
  - Increased knowledge about
    - Client needs
    - Technology capability
    - Team capability
  - Good decisions



# Then what?

- Determine the source of the variance
- Over the project lifecycle, reduce the variance to improve likelihood of shipping

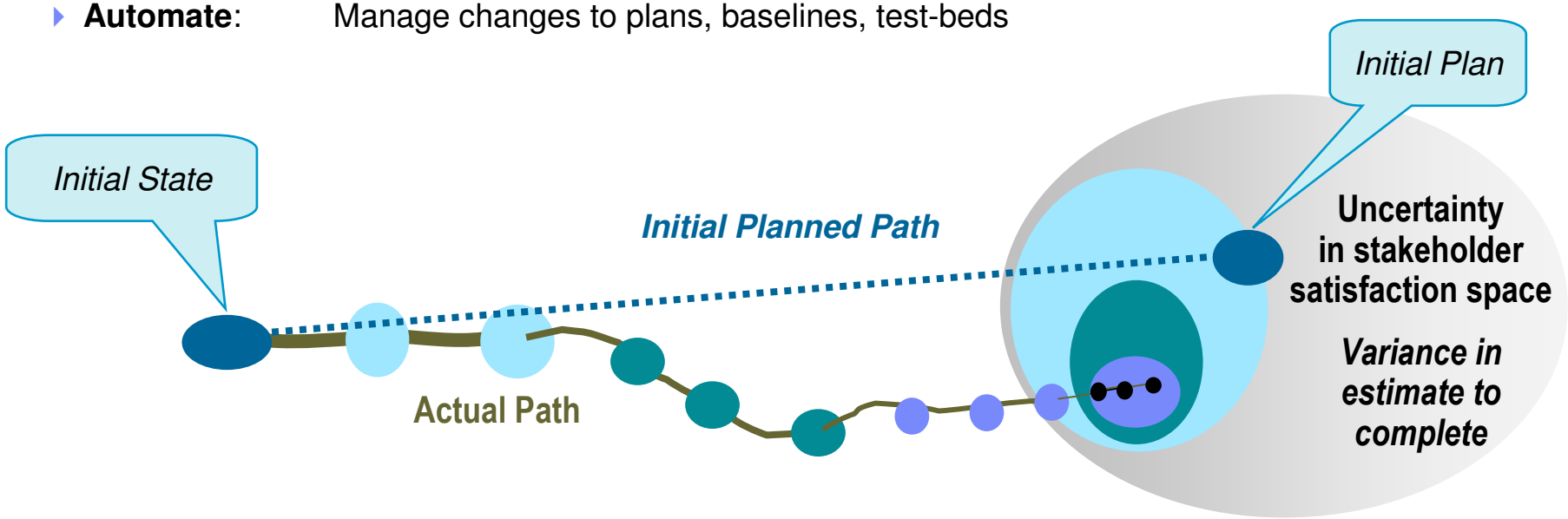


# Then what?

- Over the lifecycle, reduce the variance further to improve likelihood of shipping

# Measure and steer

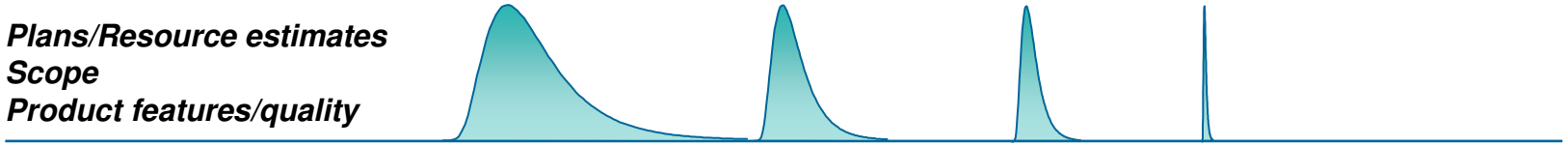
- At onset of program
  - **Report:** Establish estimates/variances of effort, cost, establish initial plan
  - **Collaborate**: Set initial scope and expectations with stakeholders
  - **Automate**: Establish a collaborative development environment
- At each iteration, improve estimates and report
  - **Report**: Values and variances of progress achieved, quality achieved, resources expended
  - **Collaborate**: With stakeholders to refine scope and plans
  - **Automate**: Manage changes to plans, baselines, test-beds



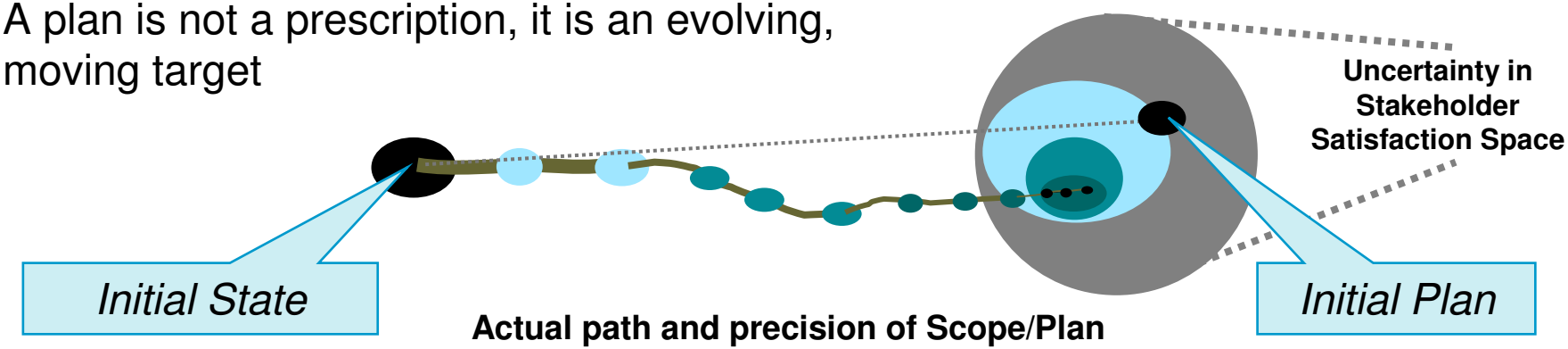
# Agile Governance = Managing Uncertainty = Managing Variance

- A completion date is not a point in time, it is a probability distribution


- Scope is not a requirements document, it is a continuous negotiation

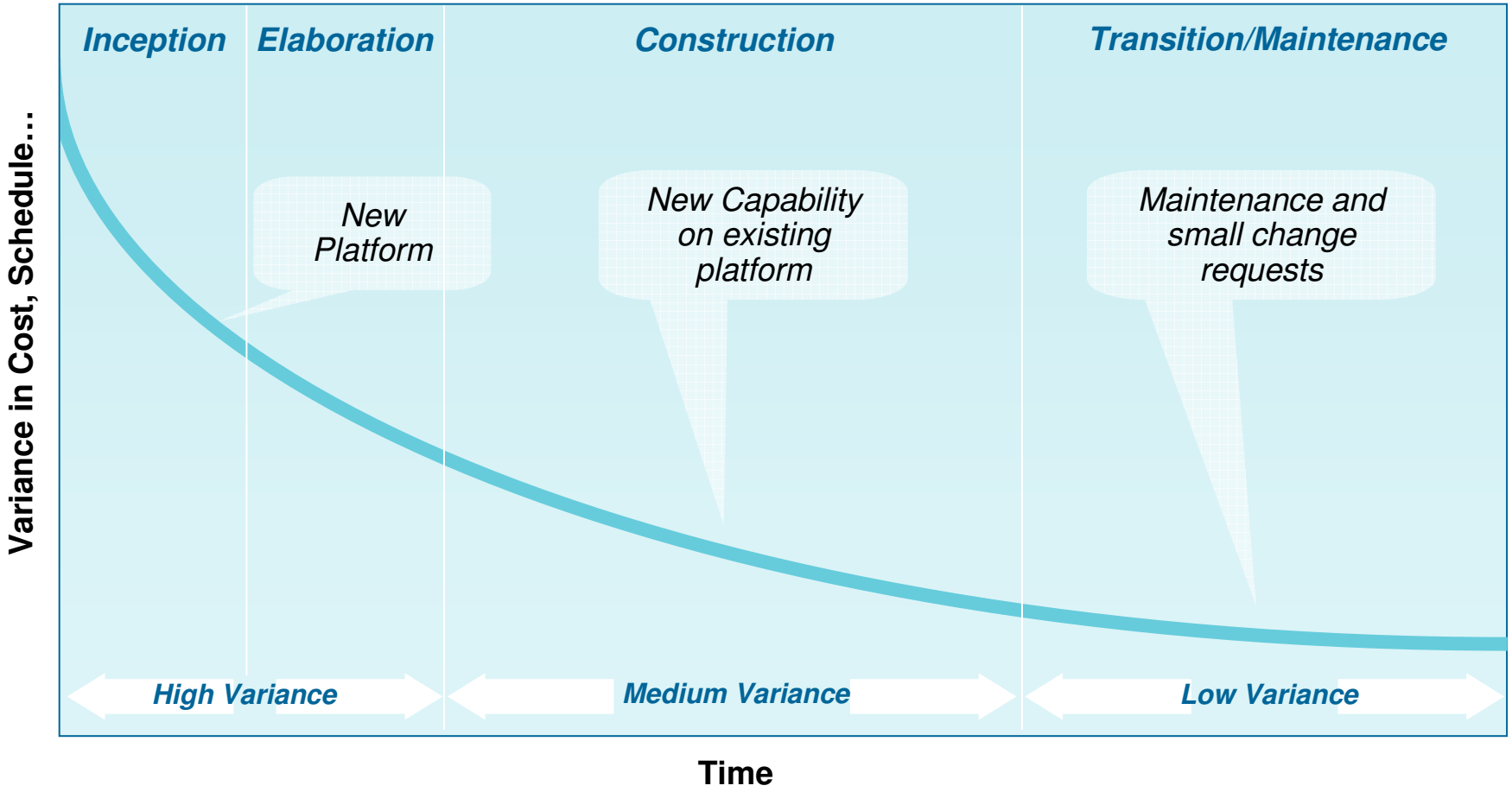



- A plan is not a prescription, it is an evolving, moving target



# Different projects need different governance

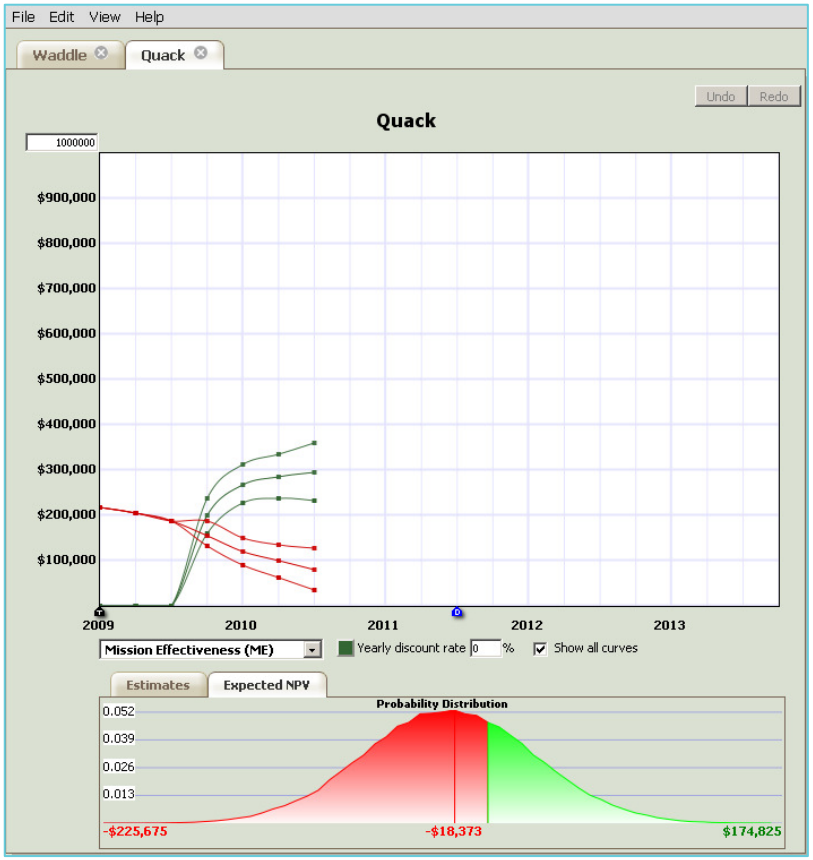
*Risk/uncertainty are the key discriminators*



# Rational is piloting two tools for managing schedule and value (benefits and cost) risk

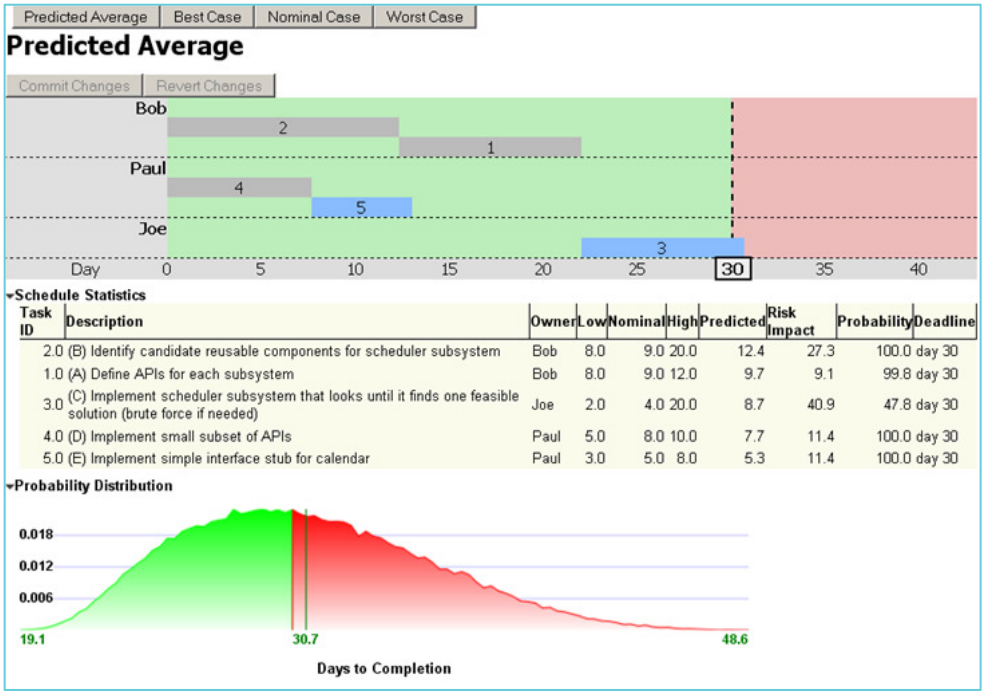
## Financier

***Ongoing view of program value***




## Tempo

***Ongoing view of schedule risk***

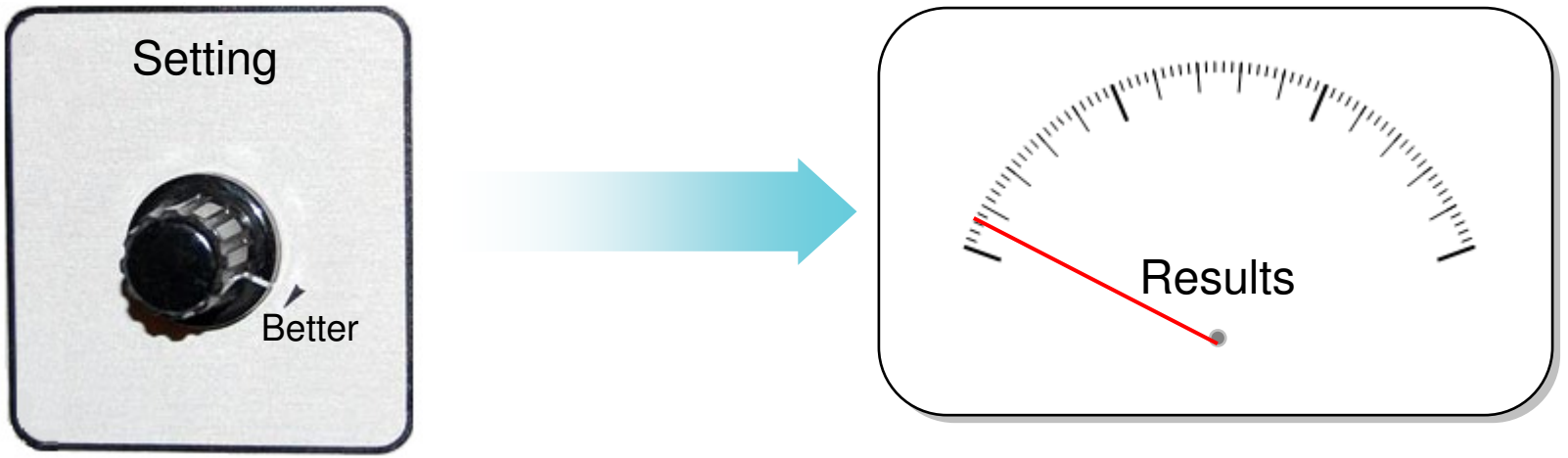


# Agenda

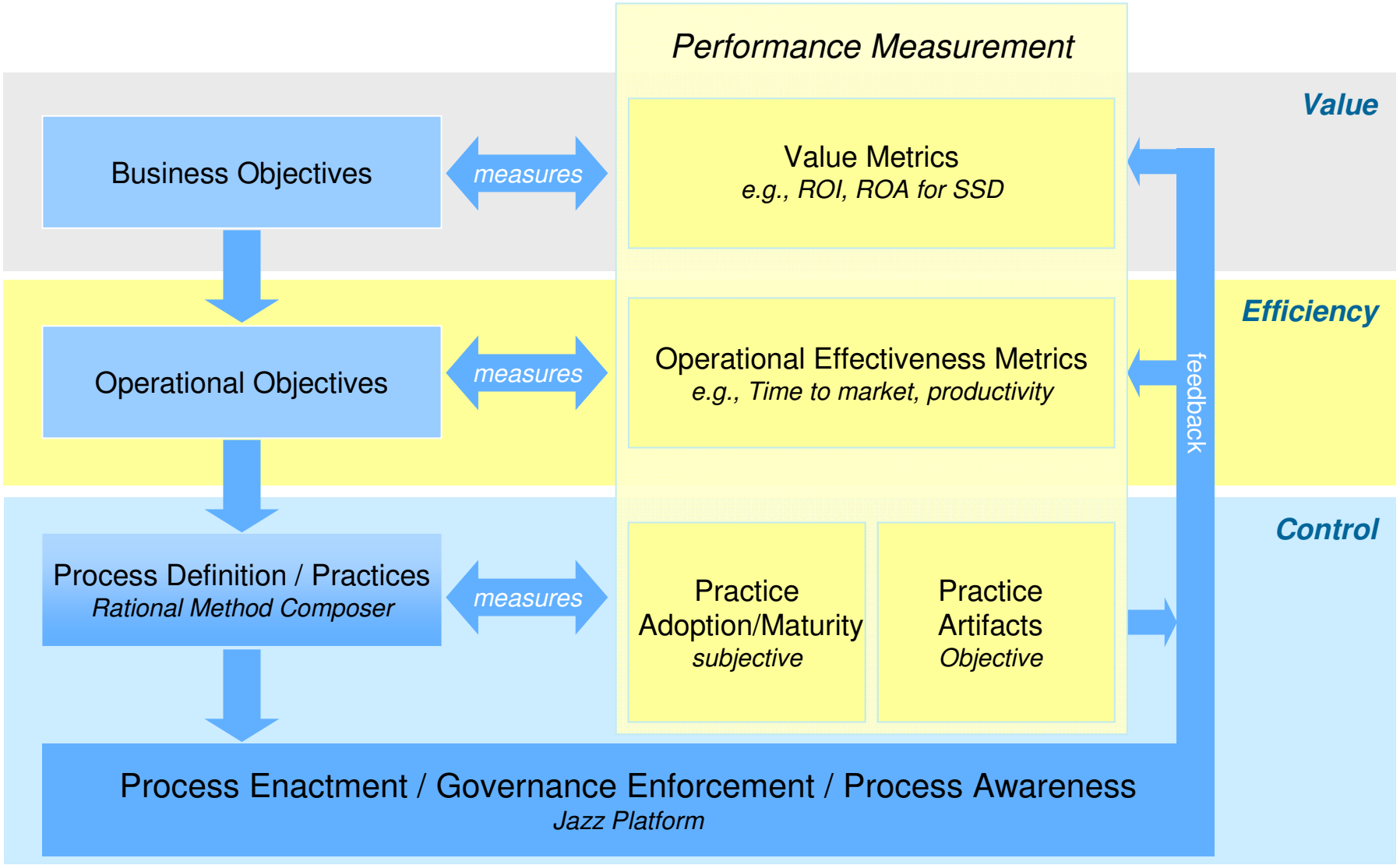
- Managing Variance: the Value of Agile Methods
- **Controlling Software Economics Through Measurement**
- IBM's Transition to Agility @ Scale
- Some final thoughts

# In a control framework, one has knobs and meters

- One sets the knobs hoping to achieve optimal meter readings
  - The meter readings are called **outcome measures**
  - Sometimes you need additional measures to ensure the system has responded to the knobs, these are called **control measures**

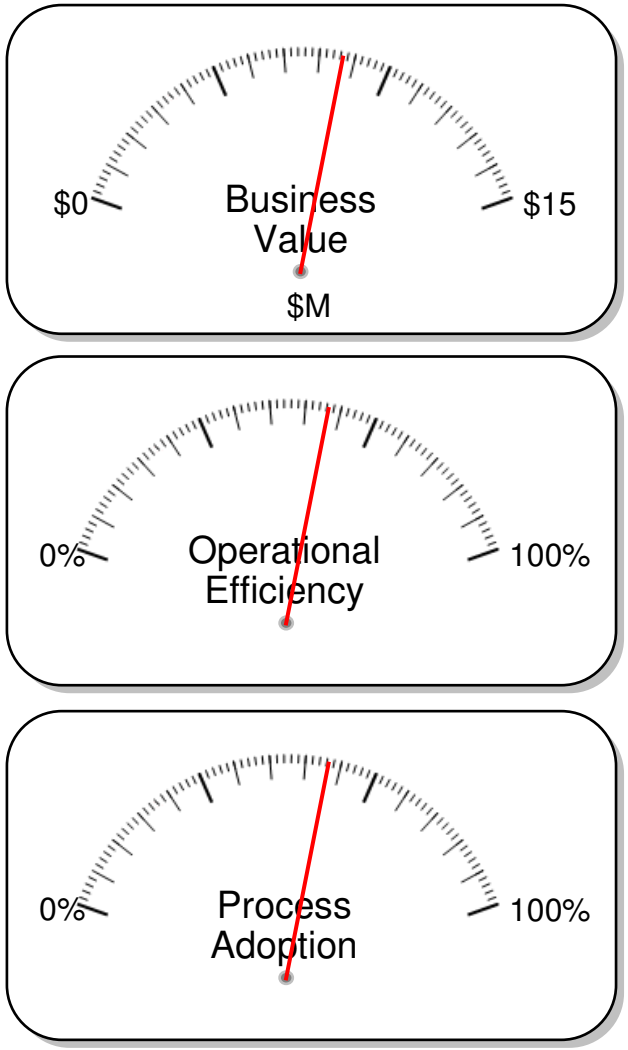


# Software and systems need a control framework



# Meters for software and systems development and delivery improvement

- Value
  - Return on Investment (ROI)
  - Return on Assets (ROA)
  - Product revenue profile
- Efficiency
  - Time to market, productivity
  - Program portfolio investment profile
  - Defect phase containment, scrap and rework rates
  - Application service levels
  - Defect densities, requirements churn, design churn
  - Skills improvement, training cost reduction
- Control
  - Practice adoption, project checkpoints
  - Artifact time between gates
  - Collaboration, skills mix



# Tailor to organizational and project context

- Agree on business value measures: Cost, profit, return on assets, market share, etc.
- Determine project mix type
  - Choose appropriate operational measures
  - Choose practices to achieve measures for project mix
  - Establish measures and feedback channels for closed loop control

| | Variance Examples | | |
|---|---|---|---|
| | ***Low*** | ***Medium*** | ***High*** |
| **Value** (Business Measures) | ▪ Cost of operations | ▪ Market share growth<br>▪ Time to market for new features | ▪ Profitability of one-of-a-kind system |
| **Efficiency** (Operational Measure) | ▪ Cost per change request<br>▪ Individual productivity | ▪ Cost per change request<br>▪ Team Productivity | ▪ Architectural stability<br>▪ Organizational productivity |
| **Controls** | ▪ Self check for practices | ▪ Beta releases<br>▪ Defect densities, removal rates | ▪ Stakeholder demonstrations |
| **Practices** | ▪ Requirements management<br>▪ Change management<br>▪ Iterative development | ▪ Agile planning<br>▪ Test driven development | ▪ Shared vision<br>▪ Risk based lifecycle<br>▪ Evolutionary Architecture |

# Measured Capability Improvement Framework (MCIF): A systematic approach to software excellence


*Determine right solution to deploy*

**Phase 1** — **Elicit and set business value objectives**

- Business objectives, contexts to operate within, and development approaches

**Phase 2** — **Determine the solution components**

- Incremental improvement roadmap
- Do Financial Analysis

*Deploy solution and measure results*

**Phase 3** — **Accelerate and monitor solution adoption using Rational's preferred approach**

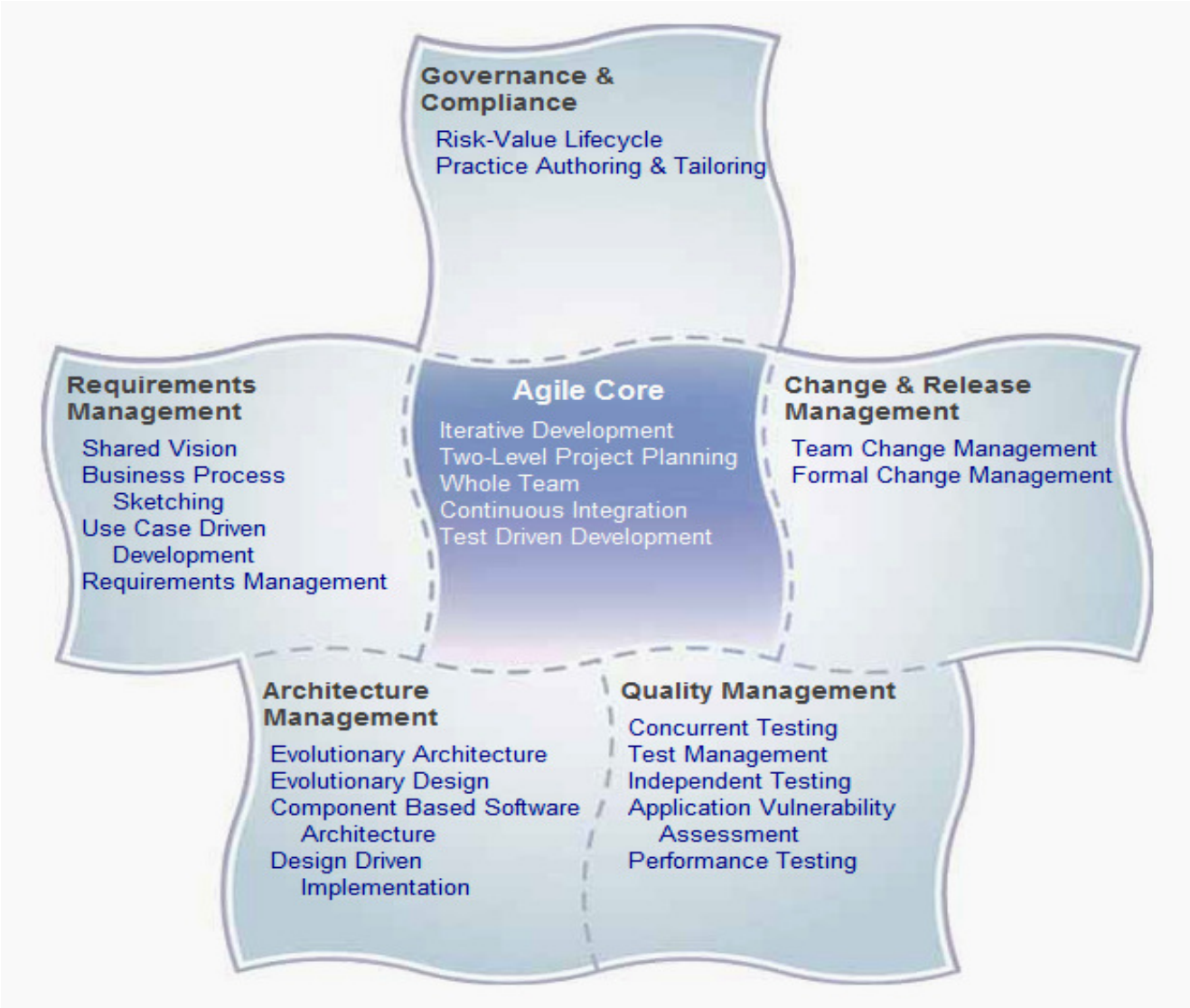
- Deploy practices and tools
- Adopt proper usage model to maximize ROI
- Measure and adapt

**Phase 4** — **Review and communicate business results**

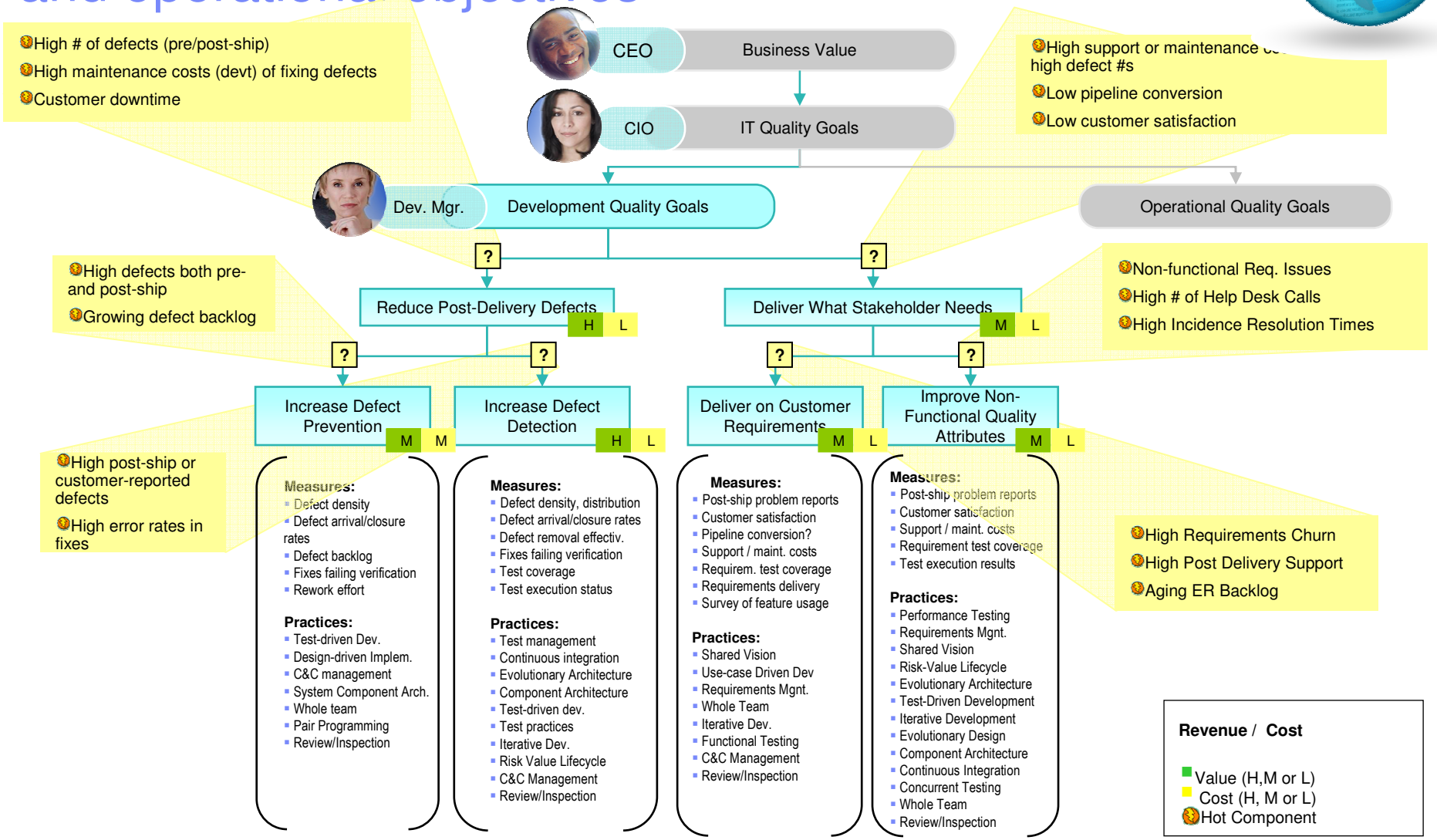
- Compile business value and results
- Conduct formal review with stakeholders

*Confirm results delivered and start on next project*

# Practices included as part of Rational Method Composer

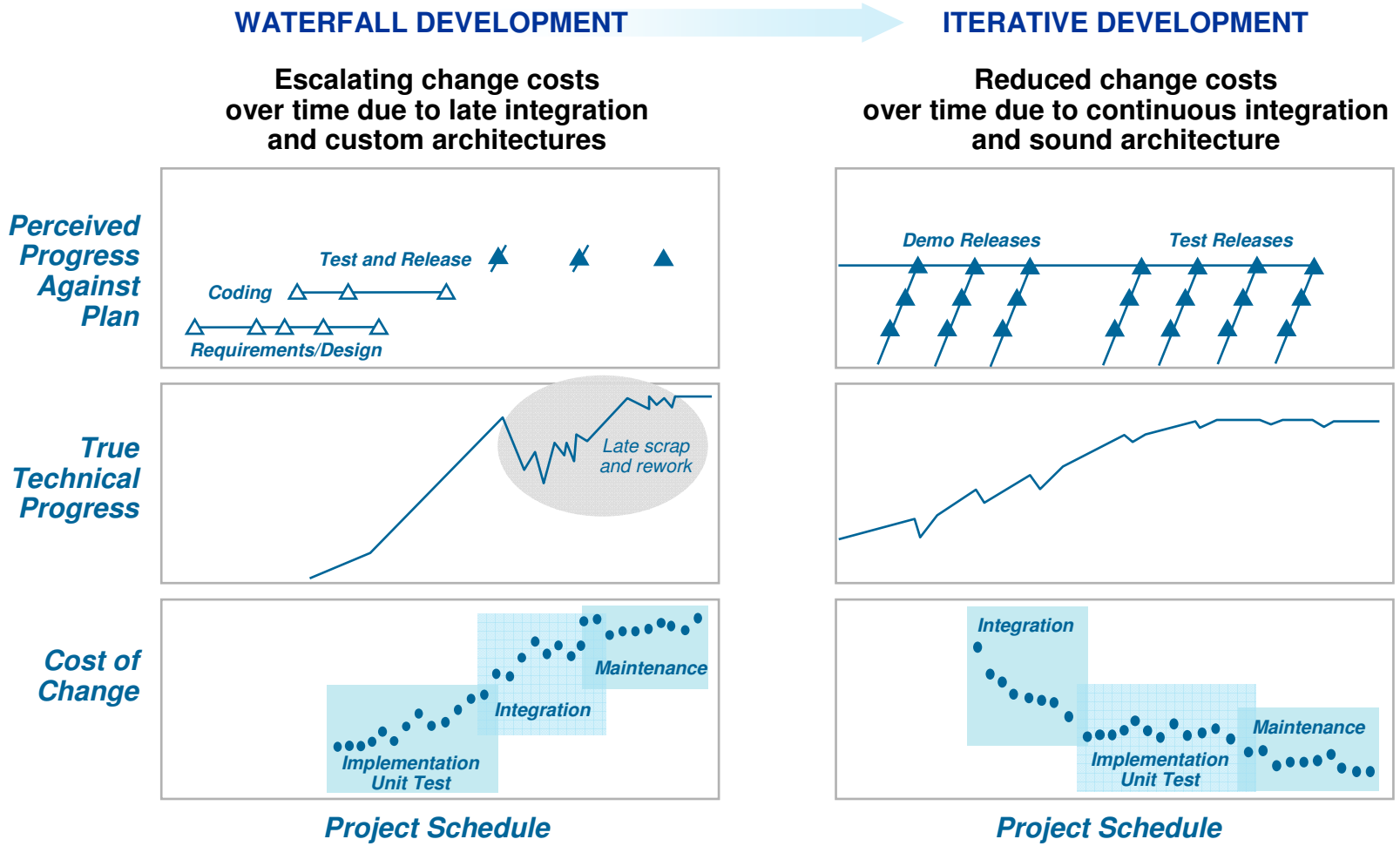



# Select practices and measures based on business and operational objectives



# Improving your process

*Measure the trends in the cost of baseline changes*

# Agenda

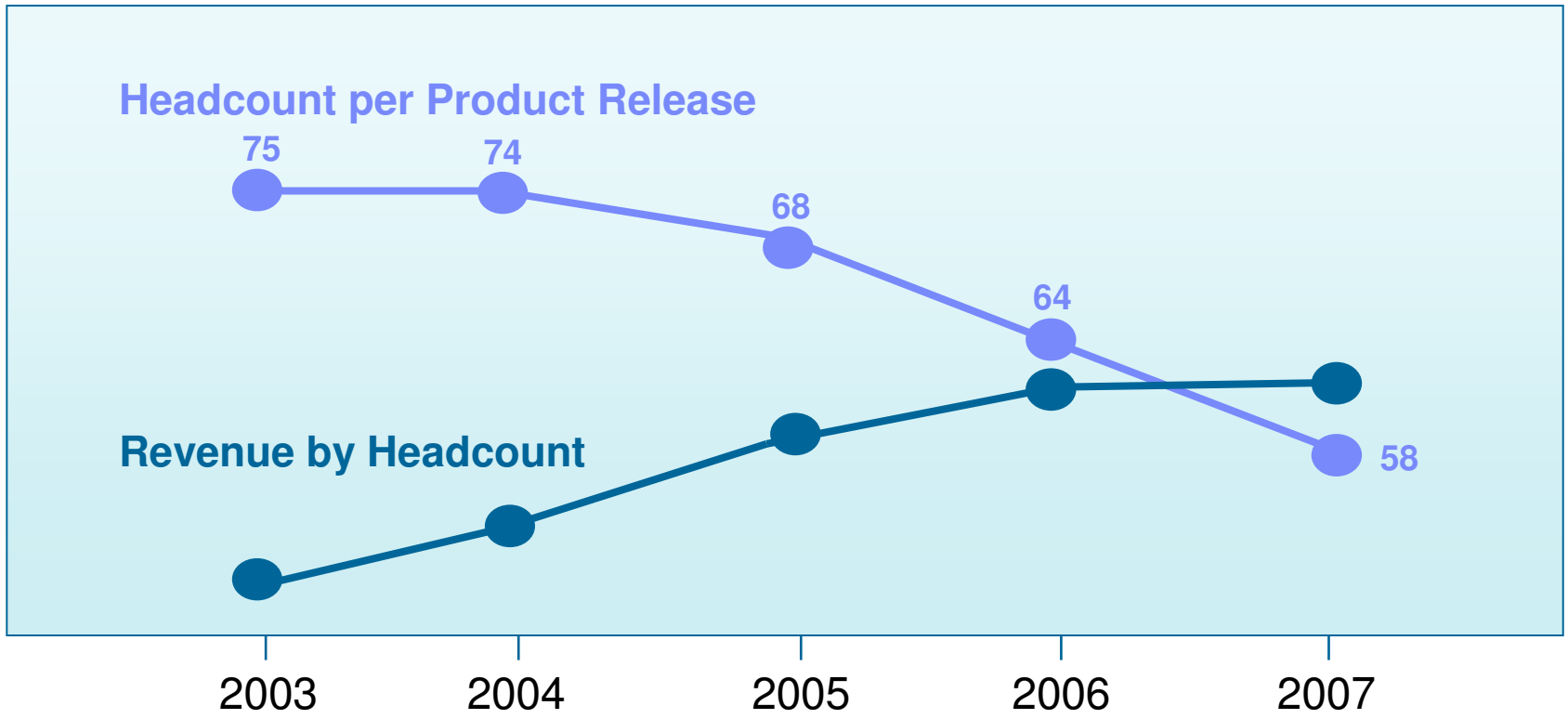
- Managing Variance: the Value of Agile Methods
- Controlling Software Economics Through Measurement
- **IBM's Transition to Agility @ Scale**
- Some Final Thoughts

# IBM SWG is Going Agile

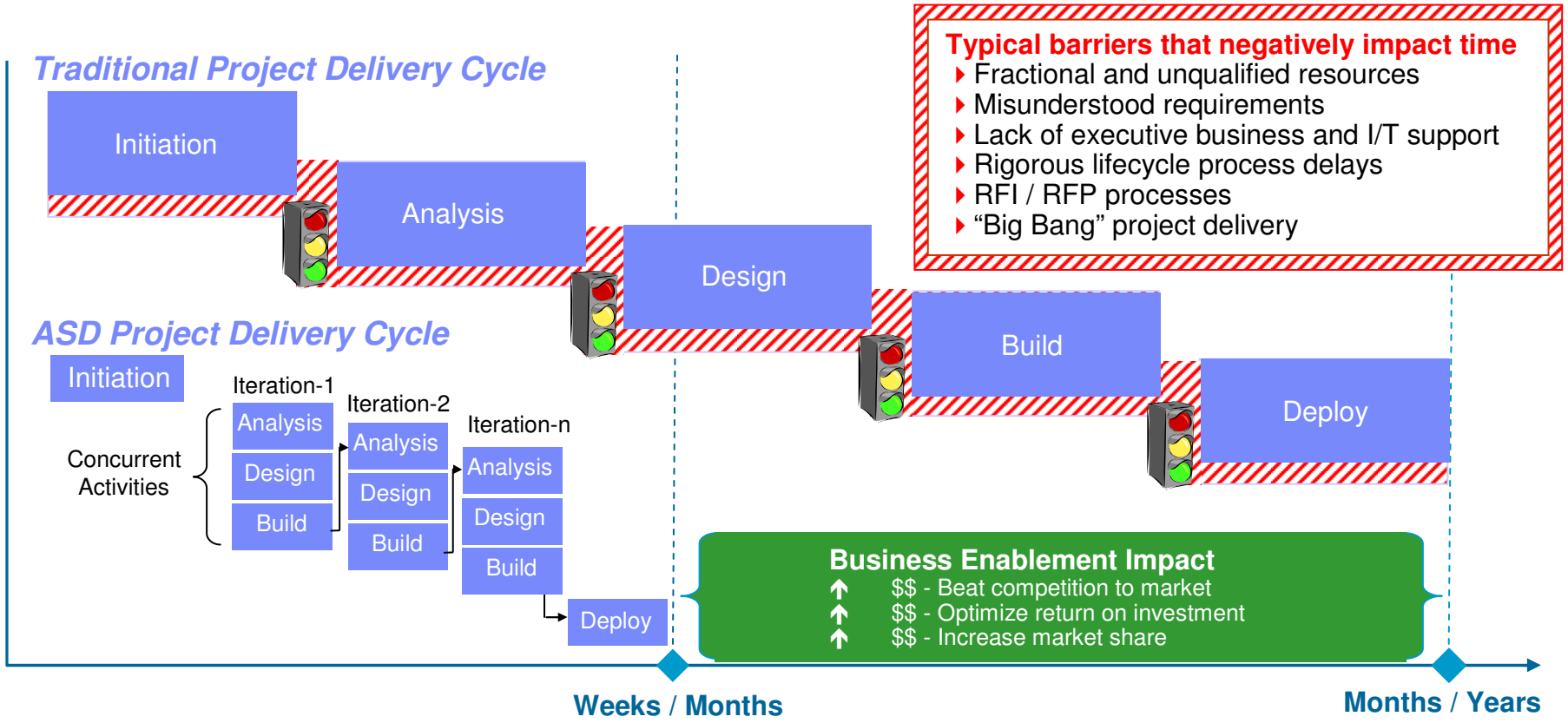
*One of the worlds largest agile transformations*

- Large scale transformation
  - ~35,000 developers
- Very diverse development contexts
  - From: New products, short time-to-market, Web 2.0
  - To: Mature products, risk reduction, older technologies
- Agility at Scale is key
  - Team size, geographical distribution, compliance, application complexity, …

# Improving productivity in IBM SWG through improved agility



# IBM GBS Accelerated Solution Delivery (ASD) Practice




**Key ASD attributes that remove or reduce time-to-market barriers**

- Right skills at right time in right ratio
- Facilitated sessions and co-location of business and I/T with onsite teams
- Organization focused only on project delivery using all best practices possible
- Agile project delivery using IBM Rational Collaborative Development Environment
- Iterative and incremental project, release and program delivery

Client Testimonial:

*"It used to take us months to just develop a business initiative and basic requirements. Now we accomplish that in days and develop the solution in months"*

# Metrics from IBM's ASD Practice accounts

| | Productivity | $'s per FP | Time to Market | Defects | Customer Satisfaction |
|---|---|---|---|---|---|
| Lincoln Financial Group | 25% | 30% | 25% | 42% | 5 of 5 |
| Scotiabank | 46% | 27% | 35% | 50% | 5 of 5 |
| RBC Royal Bank | 25% | 25% | 25% | 25% | 5 of 5 |
| Ford | 60% | 30% | 67% | To 6 per 1000 FP's | 4 of 5 |

ASD Support Material

# Agenda

- Managing Variance: the Value of Agile Methods
- Controlling Software Economics Through Measurement
- IBM's Transition to Agility @ Scale
- **Some Final Thoughts**

# Four patterns of success in achieving Agility at Scale

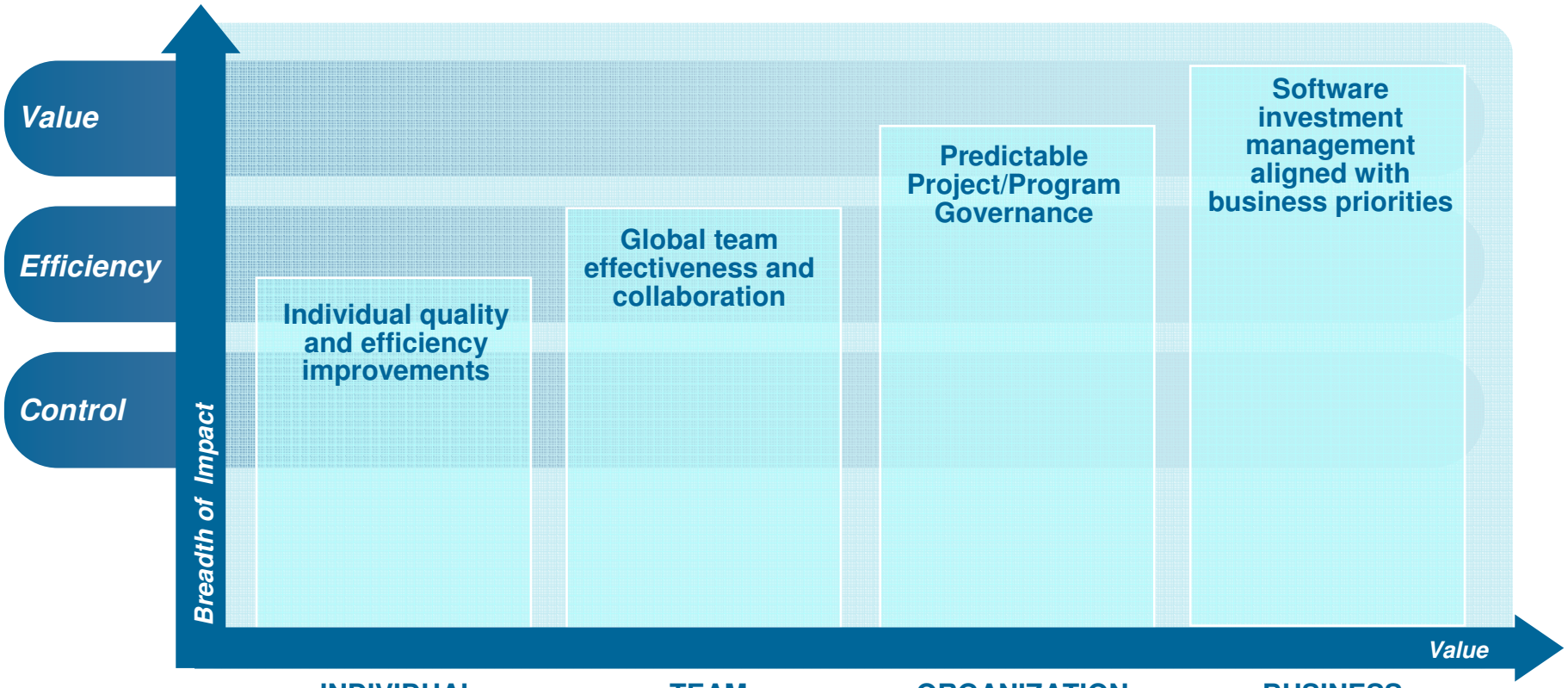
1. **Scope management ➔ *Asset based development***
   **Solutions evolve from requirements AND requirements evolve from available assets**
   ***As opposed to getting all the requirements right up front***

2. **Process management ➔ *Rightsize the process***
   **Process and instrumentation rigor evolves from light to heavy**
   ***As opposed to the entire project's lifecycle process should be light or heavy depending on the character of the project***

3. **Progress management ➔ *Honest assessments***
   **Healthy projects display a sequence of progressions and digressions**
   ***As opposed to progressing to 100% earned value with monotonically increasing progress against a static plan***

4. **Quality management ➔ *Incremental demonstrable results***
   **Testing needs to be a 1st class, full lifecycle activity**
   ***As opposed to a subordinate, later lifecycle activity***

# Effective software delivery enabled by agility and measurement



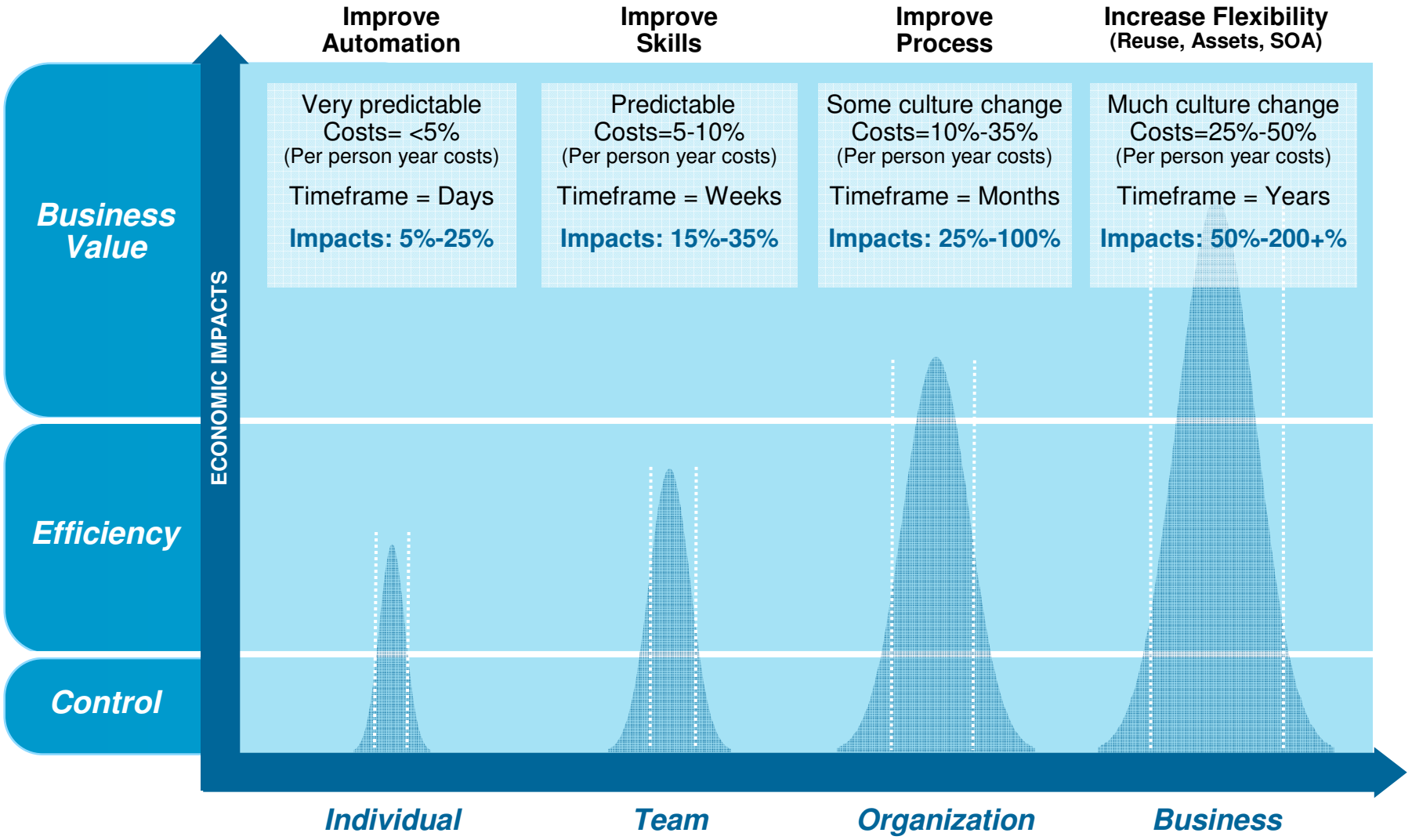

| | INDIVIDUAL | TEAM | ORGANIZATION | BUSINESS |
|---|---|---|---|---|
| ***Measures of increasing value*** | ▪ More creative time, less overhead time<br>▪ Painless governance<br>▪ More automation support | ▪ Fewer meetings<br>▪ Less scrap/rework<br>▪ Earlier defect detection<br>▪ Honest metrics | ▪ More reusable assets, services, skills, practices and measures<br>▪ More predictable outcomes<br>▪ Higher ROI | ▪ Optimized investments and supply chains<br>▪ Software development as an first class business process<br>▪ Business optimization |

# Improving software economics in each dimension

$$\text{Time or Cost To Build} = (\text{Complexity})^{(\text{Process})} * (\text{Team}) * (\text{Tools})$$

| | 1960s-1980s | 1990s-2000s | Today and forward… |
|---|---|---|---|
| **Complexity** | 100% Custom | 30% Reused Assets<br>70% Custom | **70% Reused Assets<br>30% Custom** |
| **Process** | Ad-hoc | Repeatable | **Agility at Scale<br>Managed and Measured** |
| **Teams** | Collocated<br>On the job training | Collocated<br>Software skills | **Globally Distributed<br>Skills shared anywhere** |
| **Tools** | Proprietary<br>Not integrated | Mix of Proprietary<br>& Commercial<br>Not integrated | **Commercial<br>Collaborative<br>Development Platform** |
| **Project Performance** | Predictable<br>Over budget, over schedule | Unpredictable<br>Infrequently on budget, on schedule | **Predictably<br>On budget, on schedule** |
| **Success Rate** | 10% | 25%-33% | **>60%** |

# Measuring software delivery value against cost



# Some final thoughts

***Software delivery is a discipline of software economics balancing risks and opportunities***

***Process enactment and measurement are imperatives to achieving agility at scale***

***Software delivery requires a platform that is architected for automation, collaboration and reporting***


**Learn more at:**

- IBM Rational software
- Rational launch announcements
- Rational Software Delivery Platform
- Accelerate change & delivery
- Deliver enduring quality
- Enable enterprise modernization
- Ensure Web security & compliance
- Improve project success
- Manage architecture
- Manage evolving requirements
- Small & midsized business
- Targeted solutions
- Rational trial downloads
- developerWorks Rational
- Leading Innovation
- IBM Rational TV
- IBM Business Partners
- IBM Rational Case Studies

**© Copyright IBM Corporation 2009. All rights reserved.** The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, these materials. Nothing contained in these materials is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software. References in these materials to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in these materials may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. IBM, the IBM logo, Rational, the Rational logo, Telelogic, the Telelogic logo, and other IBM products and services are trademarks of the International Business Machines Corporation, in the United States, other countries or both. Other company, product, or service names may be trademarks or service marks of others.